Lab 1

Learning Objectives

I. **How do I plan a project and execute my plan?** *How do I navigate around a Unix system?* This lab will provide the opportunity to practice navigating around a Unix system and utilizing many basic commands.
II. **Read the error output. Read it.** *Unix (and Python) will usually tell you what went wrong.* This lab will probably raise lots of errors. Instead of trying to guess where you could have possibly gone wrong, use this as an opportunity to practice directing your attention towards what Unix said went wrong.

Today we'll get comfortable working on a Unix system. You will be using the basic navigational commands, working with aliases and environment variables, and start with some basic shell scripting. These are extremely useful skills that will pay off as we begin to code on the corn system in this course.

While you work, form a habit of reading Unix's error output.
In **part 1**, you will be playing around with basic navigation in Unix and setting up a working directory for this class.
Part 2 has you write a basic shell script—a computer program to do basic file operations.
Part 3 has you examine the `~/.cshrc` file and set a few aliases and environmental variables for your later convenience.
Part 4 has you utilize your acquired Unix skills to find certain files on the Stanford AFS system.
Part 5 has some fun ways to continue learning Unix commands beyond this lab.

While You Work: Read the Error Output

Unix will throw errors. Read them in order to figure out what the error was. If you make an error and figure out what it is, let us know! We'll put it on the board as a resource for the rest of the class. We'd like to keep track of what errors you encounter so that the class as a whole can benefit.

Part 1: Basic Unix

First, log on to corn using your SSH client.

Use `ls` and `cd` to look around a bit. What is the directory structure? What is in the directories in your home directory? Remember that `.` and `..` refer to the current directory and the current parent directory, respectively. Try using `man ls` to find other options, and see if you can find any "hidden files."

Now, `cd` into a directory of your choice. Use `cd` again to go into a directory, but this time use the full path starting at `/`, not a relative path. *Hint: use* `pwd` *("print working directory") to find the current full directory path.*

Go back to your home directory (`~`). Go ahead and make a Physics 91SI directory - this will be your "working directory" for the quarter. You can call it whatever you want, and put it directly into your home directory or in a subdirectory of your choosing. *Hint: Unix is case sensitive. Also, you can't put simple spaces in names on the command line. If you want to use names with spaces in them, you can enclose your entire statement in double quotes.*

Because dealing with spaces is annoying, directory and file names in Unix typically use hyphens instead of spaces. Now enter into your working directory and make a `lab1` directory. Then make a `lab1-copy` directory by copying `lab1` (*hint: try* `man cp`):

Now get into the `lab1` directory, then use a text editor of your choice to make a text file (with ending .txt) with some content, e.g. *"Hello World!"*, inside the `lab1` directory. Without changing directories, copy that text file into your `lab1-copy` directory. Still without changing directories, read the copy of the text file on the command line to assure it has the contents that you would expect it to.

Part 2: Basic Shell Scripting

Make a new text file in your original `lab1` directory. Put in it—in order, and on separate lines—all the commands necessary to

1. Print "hello world"
2. Print the contents of the `lab1-copy` directory
3. Print the contents of the .txt file in that directory
4. Delete the `lab1-copy` directory and all its contents

Make sure to use the correct full or relative paths while writing this. Also note that relative paths are always taken relative to the current directory, which changes every time you use `cd`.

Now, figure out how to change the permissions of this text file to "executable" and do it (*Hint: try* `man chmod`).

Actually execute the file and verify that it did what it was supposed to. This is a first example of what is called a "shell script", a file that executes a fixed series of commands so that you don't have to worry about typing them over and over again. We'll talk more about these on Thursday.

Part 3 (optional): Aliases and Links

Open the file `~/.cshrc` in an editor. Where is this file located? What does the file do?

Find the relevant section, and add an alias that changes the current directory to your `lab1` directory. Try the command you created—does it work? It turns out that `cshrc` only gets invoked on login, which means that you have to log out and log back into corn for it to work or use the `source` command (see documentation).

Now try to do the same thing with an environment variable, setting its value to the full path of your Physics 91SI directory. Look up the command necessary; it should enable you to type something like `cd $SI` in your shell, as a shortcut of sorts. (*Hint:* look in the file for other examples of environmental variables)

A common alias used on many systems is `ll` as a shortcut for `ls -alF`. Set this alias up in your `~/.cshrc` file. What do all of these flags do?

You'll notice that the alias, variable, and link do more or less the same thing in this example. If you have time, look at the other aliases and variables in your `cshrc` file, and see how the link you made behaves with file system commands (`ls`, `cd`, etc.). Why is the link useful? Why might you use an environment variable or alias instead?

Part 4 (optional): Easter Egg Hunt!

There is an easter egg in one of the instructor's AFS directories. Using what you've learned about the file system and Unix commands, find the egg and run it...

Move around a bit more in the instructor folder. Can you open other files or folders? Why?

Part 5 (optional): Fun ways to continue learning Unix

Of course, there is a lot more to Unix than can possibly be covered in one lab. Should you ever find yourself wanting to familiarize yourself more with Unix commands, the following games are instructor-recommended!

1. Terminus is a text-based adventure game that teaches users how to use various terminal commands
   (http://www.mprat.org/Terminus/)
2. Bandit is another game guides you through Unix commands; the difficulty increases with each

level.

(http://overthewire.org/wargames/bandit/)